基于支点的数据中心网络地址快速自动配置方法研究

张 赣[1] 梁 伟[2] 毕经平[2] 邵定宏[1]
(南京工业大学电子与信息工程学院 南京 211800)[1] (中国科学院计算技术研究所 北京 100190)[2]

摘 要 目前云计算数据中心规模大,网络设备多,手动配置设备地址不但耗时耗力,而且容易出错。已有自动配置工作未能充分利用数据中心网络拓扑结构特征,导致从规划设计到实际设备配置的映射过程回溯步骤多,效率低。为此,提出了一种基于支点的数据中心网络地址快速自动配置方法 PFAC(Pivot-based Fast Automatic Configuration)。PFAC 通过预处理分析数据中心网络拓扑层次关系,依据拓扑特征优选支点完成快速匹配,并基于支点缩小配置映射节点的候选集,有效提高了配置效率。基于 FatTree 结构的模拟实验表明,PFAC 能够根据数据中心网络规划蓝图,自动快速地为物理设备分配地址。与经典数据中心网络地址配置方法相比,PFAC 算法平均耗时缩短了 35%。

关键词 云计算,数据中心网络,地址自动配置,基于支点,快速配置
中图法分类号 TP393.07 **文献标识码** A

PFAC: Pivot-based Fast Automatic Configuration for Data Center Networks

ZHANG Gan[1] LIANG Wei[2] BI Jing-ping[2] SHAO Ding-hong[1]
(College of Electronics and Information Engineering, Nanjing University of Technology, Nanjing 211800, China)[1]
(Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190, China)[2]

Abstract In currently, configuring addresses manually in a large-scale data center is not only time-consuming and labor-intensive but also error-prone. Related work fails to make full use of data center network topology, resulting from large number of backtracking steps unnecessarily. To solve these problems, a pivot-based fast automatic configuration algorithm(PFAC) was proposed. PFAC matchs a pair of pivots after analyzing the topology for narrowing the candidate set of the rest devices and improves the configuring efficiency. The experiments based on FatTree network topology show that the processing time of PFAC is reduced by 35% compared to the existing classical work DAC.

Keywords Cloud computing, Data center network, Address configuration, Pivot-based, Fast configuration

1 引言

云计算(Cloud Computing)[1]作为一种新兴的计算模式,近年已在科学计算、网络服务、海量数据存储等领域广泛应用,数据中心[2]作为云计算的基础设施也在被大量建设。目前流行的数据中心规模往往十分庞大,包含数万台服务器的数据中心十分常见,甚至一些大型数据中心包含了几十万台服务器。大量的服务器通过交换机连接起来,并通过标准的 TCP/IP 协议进行通信,就形成了通用的数据中心结构。

数据中心的设备需要进行必要的配置才能正常工作。出于高效路由、资源调度和容灾备份等方面的考虑,目前数据中心网络通常将网络拓扑、设备位置等信息编码到设备地址中,称之为逻辑编号。它可以是 IP 地址(例如 VL2[3]),或是 MAC 地址(例如 Portland[4]),也可以是自定义编号(例如 DCell[5] 和 BCube[6])。在配置阶段,就需要为每个设备分配相应的逻辑编号。

对于小规模数据中心,可以通过在 DHCP[7] 服务器中手工添加静态映射关系表的方式来解决逻辑编号的分配问题。但是目前的云计算数据中心设备数量巨大,手工静态配置的方式不但工作量繁重,而且不可避免地会出现很多错误。据统计[8],57%的数据中心故障是人工操作失误所造成的。为了解决以上问题,Kai chen 等人在 ACM SIGCOMM 2010 中提出了一种基于同构图性质的地址映射算法(DAC[9])来自动地为设备分配地址。但是,由于 DAC 没有充分考虑数据中心网络结构的特殊性和设备之间的互联关系,导致映射过程中回溯步骤过多,影响算法效率。

本文提出了一种基于支点的数据中心网络地址快速自动配置算法 PFAC(Pivot-based Fast Automatic Configuration)。该算法充分考虑数据中心中网络拓扑的特殊性,基于网络层次结构寻找根节点设备并快速完成地址分配,再以此为支点,依据拓扑连接关系完成其他节点的配置,以达到缩短回溯步骤、提高算法效率的目的。在基于仿真的验证实验中,PFAC 算法执行效率比已有经典算法 DAC 提升了 35%。

本文第 2 节介绍相关的工作,并进行了对比;第 3 节对数据中心地址自动配置进行了形式化建模;第 4 节详细阐述了 PFAC 算法;第 5 节通过仿真实验将 PFAC 与已有经典方法

到稿日期:2012-07-24 返修日期:2012-11-30 本文受国家重点基础研究发展计划(973 计划)(2011CB302505)资助。
张 赣(1987—),男,硕士,主要研究方向为云计算网络优化配置,E-mail:zgwonder@gmail.com;梁 伟(1983—),男,博士,主要研究方向为网络测量和性能分析;毕经平(1974—),女,教授,博士生导师,主要研究方向为下一代互联网、网络监测与管理、网络行为分析;邵定宏(1951—),男,教授,硕士生导师,主要研究方向为智能计算。

进行对比，并对结果进行分析；最后对本文工作进行了总结。

2 相关工作

目前，关于通用高效的数据中心地址自动配置的研究比较少，原因在于目前数据中心网络组建还未形成一个认可度较高的规范，现有的自动配置算法都是针对于一些特定的网络拓扑设计的。

在 ACM SIGCOMM 2009 中，R. N. Mysore 等人针对数据中心网络的特点，提出了 Portland 拓扑结构。它可以依据设备的 MAC 地址，使用分布式位置发现协议（LDP）来确定设备位置。但是，LDP 只适用于多层次树形网络拓扑结构，而无法保证在其他网络拓扑结构下的适用性。

目前，在数据中心组网过程中，广泛采用以太网技术。在以太网中，网桥可以通过广播报文的方式自动学习网络中的拓扑关系，但是受制于规模，这种方法很难适用于云计算数据中心的地址配置[10-12]。为了使这种网络拓扑学习方法能适用于规模较大的以太网环境，C. Kim 等人提出了 SEATTLE[13] 配置算法，其使用单跳 DHT 来分配交换机中的 ARP 状态，从而达到获取网络拓扑信息、设备位置关系等目的。但是 SEATTLE 仍需要交换机广播报文，每台交换机的负载会随着终端主机的数量而增加，这种需要发送广播报文的学习方式仍然很难适用于数据中心网络环境。

Kai Chen 等人提出了一种基于子图同构（GI）的映射算法 DAC，它能适用于各种规模、各种拓扑结构的网络环境，自动为设备配置地址。但是 DAC 并没有充分利用数据中心网络拓扑特性，导致映射回溯步骤过多、效率较低。

3 问题描述与建模

如图 1 所示，PFAC 的输入信息包括数据中心网络规划蓝图和物理网络拓扑图。数据中心网络规划蓝图包含了每个节点的逻辑编号、设备类型以及节点之间的相邻关系等信息，由数据中心网络设计人员事先给出；物理网络拓扑图记录了数据中心网络的拓扑结构以及设备的唯一性标识（我们称之为物理编号），可通过拓扑发现获得。数据中心设备逻辑地址自动配置就是要实现网络规划蓝图中的逻辑编号到物理网络拓扑图中的物理编号的一一映射，并基于映射关系为每个物理设备分配逻辑编号。

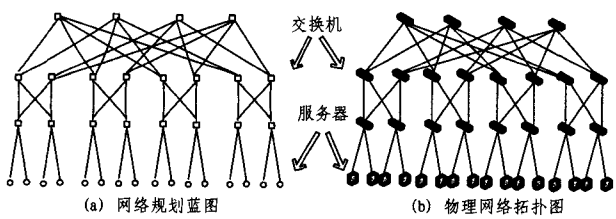

图 1 数据中心网络规划蓝图与物理网络拓扑图

我们运用图论的相关知识对问题进行建模：给定蓝图 $G_b=(V_b,E_b)$，物理拓扑图 $G_p=(V_p,E_p)$。顶点集 V_b 与 V_p 代表图中的设备节点，边集 E_b 和 E_p 代表图中设备的互联关系。为满足地址分配要求，我们需要寻找节点集 V_b 到 V_p 的一一映射，同时保留 E_b 与 E_p 的对应邻接关系，这正是典型的两图同构（GI）[14] 问题。根据图的同构的相关性质，若 G_b 与 G_p 同构，则可基于同构双射函数 f 构建顶点的一一映射。反之，则说明 G_b 与 G_p 不同构，网络规划蓝图的设计与实际网络拓扑结构不符，无法为网络设备分配逻辑编号。

4 基于支点的快速自动配置算法

为了实现逻辑编号和物理编号的快速匹配，提出了一种基于支点的快速自动配置算法 PFAC。它首先基于经典同构判定方法（Saucy[15]）提出一个节点匹配判定基础算法，再根据网络结构特点对全图映射过程进行优化，以减少回溯步骤，提高算法效率。

4.1 节点匹配判定基础算法

Saucy 是一个经典的同构判定算法，在数字设计自动化领域广泛应用于寻找图中自同构节点的映射关系。我们抽取 Saucy 中对于两节点是否能匹配成功的判定方法应用于 PFAC 中，其具体算法如下：

假设在图 $G=(V,E)$ 中，V 是 G 中所有顶点的集合，V 由一些不相交的非空子集 $\theta^i(0<i\leqslant n)$ 组成，定义 3 种针对图 G 的操作。

操作 1 抽出（decompose）：图 G 中存在一个节点 v，和一个 V 的子集 θ^i，$v\in\theta^i$，$\theta^i\subseteq V$，将 v 从 θ^i 中抽出等于将元素 v 从 θ^i 中取出，独立形成一个新的集合 $\{v\}$，并将 $\{v\}$ 加入 V 中。

操作 2 分解（split）：在图 $G=(V,E)$ 中存在 V 的子集 θ^1,θ^2。用 θ^1 分解 θ^2 意味着对所有 θ^2 中的元素计算在 θ^1 中与它有连接的顶点的个数 k，并将 θ^2 中的元素按照 k 的大小划分成若干子集。

操作 3 重构（refinement）：该操作将抽出与分解组合使用，在对图 $G=(V,E)$ 中的某一顶点进行抽出操作后，使用新生成的子集依次分解其他子集，当有新的子集产生时，继续对其执行分解操作，直到 V 中不存在能被分解的子集时结束。称图 G 此时处于稳定状态。

如果两个图中的顶点集都处在稳定状态，且顶点集中子集个数一致，位置对应的顶点子集中元素个数都相等，则称两个图的顶点集同型（apposition）。若一个顶点所在的子集中只有这个顶点本身，则称这个顶点处于独立状态（singleton）；当顶点集中所有的顶点都处于独立状态时，称这个顶点集是离散的（disperse）。

对两个顶点 v_b 与 v_p 能否形成映射进行判断，首先在当前节点集 V_b 与 V_p 中对 v_b 与 v_p 进行抽出，完成后重构节点集 V_b 与 V_p，若重构完成后 V_b 与 V_p 同构，则说明 v_b 与 v_p 可以形成映射，进一步对剩下的节点进行匹配；反之，说明 v_b 与 v_p 无法映射在一起，则回溯寻找新的节点进行匹配。若剩下的节点无法在图中找到能与之匹配的元素，那么必须对已经形成映射的结果进行回滚，直到顶点集处于离散状态（说明已经找到两图节点的一一映射）或回滚到根节点（说明两图不同构）为止。

当图中顶点数量 n 与节点最大度数 d 确定时，Saucy 图同构判定算法的时间复杂为 $n^{O(d^2)}$。由于现有数据中心网络中设备数量往往较大，Saucy 算法很难直接应用于数据中心设备地址配置，因此 PFAC 基于网络结构的特点对全图映射过程进行了优化。

4.2 全图映射优化算法

经典地址配置方法 DAC 对 Saucy 提出了基于节点间最短距离矩阵的优化算法，以提高配置效率。但是由于 DAC 未能充分利用顶点间的拓扑关系来缩小候选集，导致算法执行

过程中由于匹配失败而造成的回溯过多，算法效率还有进一步提高的空间。为此，PFAC 提出了一个基于支点，并充分利用网络结构特殊性的优化算法。它利用网络核心层设备数量较少的特点优选支点快速匹配，并依据同构图的边与节点映射关系理论，基于支点连接关系缩小候选集，减少回溯次数，加速匹配剩余节点的过程，提高节点映射效率。

如图 2 所示，PFAC 通过预处理分析数据中心网络拓扑层次关系，在蓝图中选择一个层数最高的节点作为初始支点，接着在物理拓扑图中圈出同样处在最高层的节点组成候选集。数据中心网络的核心层设备数量较少的特点保证了候选集规模较小，使其可以快速在候选集中找出能与支点形成映射的节点，组成初始支点对；接着根据节点间相邻关系，以初始支点为起点在网络规划蓝图中进行遍历搜索。每遍历到一个新的节点，便根据其与支点的连接关系，在物理拓扑图中圈定出新的候选集，找出能与之匹配的节点形成映射关系，并形成新的支点对，为匹配后续遍历到的节点提供支持，直到所有节点形成一一映射为止。

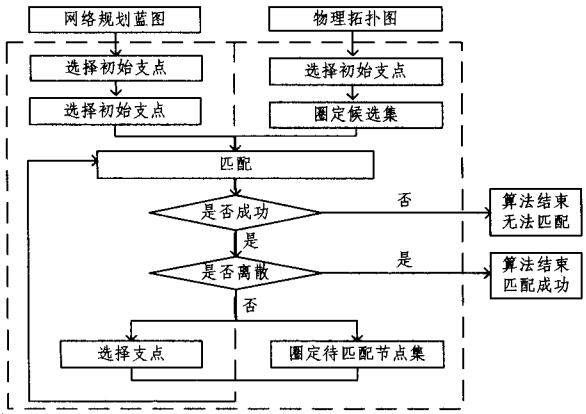

图 2　PFAC 算法流程图

4.2.1　预处理

预处理过程中，PFAC 对设备层次的划分基于拓扑发现的结果，根据设备节点的位置与类型来确定设备所在层数，具体算法如下：

1）首先将图中所有节点的层数置 0 后装入同一个集合，构成未分层集合，接着在该集合中查找设备类型为服务器的节点并从此集合中取出，构成一个新的集合，称之为参考集合；

2）将未分层集合中的所有节点层数加 1，并取出其中与当前参考集合中的节点有连接关系的部分，构成新的参考集合；

3）重复执行步骤 2）与步骤 3），直到未分配集合为空。这时设备层次划分算法结束，所有节点的层次已被分出。算法伪代码如下：

```
Layer Finding(V) //初始化，以图中节点的集合作为输入
  foreach vertex v∈V
    If(v. type==server)
      {delete v from V;
      add v into reference;}
  foreach vertex v∈V
    v. layer++; //所有设备层数加 1
  While(true) {
    if(Device is empty) {return true;}
    foreach vertex v∈V
      foreach vertex v′∈reference
        if(v connect with v′)
          {delete v from V;
          add v into new reference;}
  foreach vertex v∈V
          v. layer++;
  }
```

DAC 基于拓扑图中每个节点到达所有其他节点的最短路径分布值（SPLD）进行优化，而当图中节点数量特别庞大时，SPLD 的运算量也非常庞大；PFAC 的层次划分算法的时间复杂度为 $O(nm)$（n 代表设备总数，m 代表网络最大层数），而现有数据中心网络层数通常较小，因此 PFAC 预处理算法的时间复杂度较低。

4.2.2　节点匹配

DAC 对节点的匹配顺序比较随机，因此当一对顶点出现映射错误时，往往需要回滚多步才能完成修正，这一过程会占用大量的算法执行时间。PFAC 对顶点的匹配顺序遵循以支点为起点的深度遍历原则，当匹配错误发生时，能在最短的时间内发现错误，及时回滚，有效地提高了匹配速度。

为方便讨论，首先定义引理如下。

引理 1　若 $G_1 \subseteq G_2$，对于 $V_i, V_j \in G_1, V_m, V_n \in G_2$，存在双射函数 $f: V_i \to V_m$，使得 $f(V_i)=V_m$，$f(V_j)=V_n$，如果 V_i 与 V_j 相连，则 V_m 必然与 V_j 相连。

引理的证明依据图同构的必要条件为：同构的两个图的节点之间具有保持相邻关系的一一对应。

完成设备层次划分后，首先判断网络规划蓝图的最高节点层数与物理拓扑图中节点的最高层数是否相同，若不同，则说明两图不同构，PFAC 算法结束；若相同，则在蓝图中选择一个层数最高的节点作为支点，并在物理拓扑图中圈出所有层数最高的节点组成候选集。接着利用节点匹配判定基础算法在候选集中找出能与支点形成映射的节点，组成初始支点对。

接着对剩余节点进行映射。为了保证每个节点都能被匹配到，我们采用图的深度遍历算法来寻找节点，在保留节点间位置关系信息的同时，保证节点映射的完备性。根据引理 1，可以使用节点相连关系来缩小每个节点匹配的候选集，并尽早发现匹配出现错误的情况，及时回滚，加速形成一一映射的过程。具体过程如下：

存在规划蓝图 $G_b=(V_b, E_b)$ 和物理拓扑图 $G_p=(V_p, E_p)$，并以 (p_b, p_p) 作为初始支点；

1）以 p_b 为起点深度优先遍历网络规划蓝图 G_b，将遍历到的第一个新节点记为 v_b'；

2）基于初始支点对，在物理拓扑图 G_p 中找出与 p_p 有连接的节点，并组成新的候选集 V_m；

3）根据节点匹配基础算法，在 V_m 中寻找能与 v_b' 形成映射的节点 v_p'，若遍历 V_m 后无法找到能与 v_b' 形成映射的节点，则 PFAC 算法终止；

4）继续深度优先遍历网络规划蓝图 G_b，记录遍历到的第一个新节点，在 G_p 中寻找与 v_p' 有连接的节点，组成候选集，然后在其中寻找节点与 G_b 中新遍历到的节点形成映射，并将 v_p' 更新为该节点，若无法找到，则 PFAC 算法终止；

5）反复执行过程 4），直到两图中所有节点都处于独立状

态时，PFAC算法结束，设备逻辑编号自动分配完成。

具体算法伪代码如下：

```
Rest_Mapping(Gb,Gp,pb,pp) //以初始支点对作为输入
if(Gb,Gp is singleton)
  return true;
θ=connect(pp);//θ是与 pp 有连接关系待匹配的节点集
w=FristConnectVertex(pb);//w 是通过 pb 遍历到的第一个节点
while(w>0){
  v=Base_Node_Mapping(w,θ);//在 w 中寻找能与 θ 形成映射的节点 v
  Rest_Mapping(Gb,Gp,w,v);
}
```

由于PFAC利用支点连接关系，把匹配一个节点的候选集缩小至与对应支点有连接关系的节点集合规模，因此减少了回溯步骤，缩短了出错时回滚的步长，PFAC算法的时间复杂度是$O(dn)$（d代表节点最大度数，n代表节点数量），与DAC的时间复杂度$O(n^2)$相比，由于在数据中心中节点最大度数通常远小于节点数量，因此PFAC的执行效率要高于DAC算法。

5 算法仿真结果与分析

为了测试PFAC算法对数据中心地址配置效率的优化效果，对PFAC进行了模拟试验，并将其与传统经典算法DAC的实验结果进行了对比分析。

5.1 实验设置与环境

选择目前数据中心最常见的两种拓扑结构FatTree与VL2进行模拟实验。FatTree多用于中小规模的数据中心网络环境，而VL2因其出色的扩展性在大规模数据中心中应用广泛。由于实验条件的限制，在单台服务器上对PFAC进行了模拟。首先使用邻接表的方式生成并存储网络规划蓝图与物理拓扑图，然后使用Java模拟的PFAC核心程序（预处理与节点匹配）完成网络地址自动配置。为了测试PFAC在不同规模数据中心中的表现，模拟如表1所列的网络环境，并选择DAC为参照系对PFAC进行了对比实验。

表1 测试环境设备数量表

FatTree(n)	VL(n,m)
FatTree(20)=2500	VL(20,100)=52650
FatTree(40)=18000	VL(40,100)=102650
FatTree(60)=58500	VL(60,100)=152650
FatTree(80)=136000	VL(80,100)=202650

所有模拟程序运行在一台CPU为Intel Core i3 M330、内存2.00GB的服务器上。选取以下两个指标对比分析PFAC与DAC：

1）配置映射回溯次数：减少回溯次数是提高算法效率的关键。在模拟程序中记录发生回溯的次数，并比较PFAC与DAC的实验结果。

2）配置映射总时间：测量在上述结构中PFAC算法与DAC算法的执行时间，直观地考察二者的执行效率。

5.2 实验结果分析

5.2.1 配置映射回溯次数

如图3所示，PFAC的匹配回溯次数明显少于DAC。在FatTree拓扑结构中，PFAC能比DAC减少43.7%的回溯次数。例如在FatTree(20)中，DAC需要产生回溯405450次，而PFAC仅需要203260次；在VL2拓扑结构中，PFAC能比DAC减少39.8%，在规模最大的VL(80,100)中，DAC需要产生回溯近993万次，而PFAC仅需要672万次。因此，总体来说，PFAC回溯次数平均能比DAC减少42%。

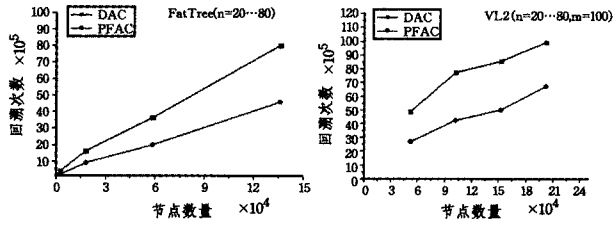

图3 PFAC与DAC的回溯次数对比图

5.2.2 配置映射总时间

如表2所列，PFAC相较DAC，在执行时间上有明显的优势。例如，在大型网络VL(80,100)中，PFAC能在35s内完成逻辑地址分配，而DAC需要53.7s；在小规模网络FatTree(20)中，PFAC与DAC相比，也能提高配置效率29%。因此，实验结果证明，相较DAC，PFAC平均能提升配置效率35%。

表2 PFAC与DAC执行时间表

网络拓扑结构		DAC运行时间(s)	PFAC运行时间(s)
FatTree(n)	n=20	1.7	1.2
	n=40	6.8	5.0
	n=60	16.0	11.0
	n=80	40.1	29.2
VL(n,100)	n=20	24.3	15.4
	n=40	37.6	21.3
	n=60	43.2	25.7
	n=80	53.7	34.1

结束语 地址配置是云计算数据中心网络运维管理中的一项重要工作。本文针对当前数据中心网络地址配置繁琐且容易出错，已有自动化配置方法回溯步骤多、效率低的问题，提出了一种基于支点的数据中心网络地址快速自动配置算法PFAC。该算法依据数据中心中网络拓扑的特殊性，优先选择上层设备节点快速完成映射，然后再以此为支点，利用网络设备之间的互联关系匹配其他节点。与原有经典方法DAC相比，PFAC算法的平均配置时间缩短了35%。

参 考 文 献

[1] 李乔，郑啸. 云计算研究现状综述[J]. 计算机科学，2011，38(4)：32-37

[2] 刘晓茜. 云计算数据中心结构及其调度机制研究[D]. 合肥：中国科学技术大学，2011

[3] Greenberg A, Jain N, Kandula S, et al. VL2: A Scalable and Flexible Data Center Network[C]// Proc of SIGCOMM 2009. NJ: ACM, 2009: 51-62

[4] Mysore R N, Pamboris A, Farrington N, et al. PortLand: A Scalable Fault-Tolerant Layer 2 Data Center Network Fabric[C]// Proc of SIGCOMM2009. NJ: ACM, 2009: 39-50

[5] Guo C, Wu H, Tan K, et al. DCell: A Scalable and Fault Tolerant Network Structure for Data Centers[C]// Proc of SIGCOMM 2008. NJ: ACM, 2008: 75-86

[6] Guo C, Lu G, Li D, et al. BCube: A High Performance, Server-centric Network Architecture for Modular Data Centers[C]// SIGCOMM 2009. NJ: ACM, 2009: 63-74

[7] 贾小东,孙向辉,彭四伟. DHCP 协议缺点及其解决方案[J]. 计算机工程,2007,23:138-139

[8] Kerravala Z. As the value of enterprise networks escalates, so does the need for configuration management[R]. Boston: The Yankee Group,2004

[9] Chen Kai, Guo Chuan-xiong, Wu Hai-tao, et al. DAC: Generic and Automatic Address Configuration for Data Center Networks [C]//Proc of SIGCOMM 2010. NJ: ACM,2010:84-99

[10] Rodeheffer T, Thekkath C, Anderson D. Smart Bridge: A scalable bridge architecture[C]//Proc of SIGCOMM 2000. NJ: ACM 2000:201-211

[11] Myers A, Ng E, Zhang H. Rethinking the service model: scaling Ethernet to a million nodes[C]//Proc of Hot Nets 2004. NJ: ACM,2004:87-100

[12] Perlman R. Rbridges: Transparent routing[C]//Proc of Infocom 2004. NJ: IEEE,2004:105-118

[13] Kim C, Caesar M, Rexford J. Floodless in SEATTLE: a scalable Ethernet architecture for large enterprises[C]//Proc of SIGCOMM 2008. Vol 29, NJ: ACM, February 2011

[14] 解春欣,汪卫. 子图同构验证算法 OES [J]. 计算机工程,2011,3:74-78

[15] Darga P T, Sakallah K A, Markov I L. Faster Symmetry Discovery using Sparsity of Symmetries[C]//45st Design Automation Conference. 2008

(上接第 53 页)

错误模型及 GSPN 模型。

```
error model CPUEM
features
  Error_Free:initial error state;
  TempErr:error state;
  PermErr:error state;
  ErrND:error state;
  Failed:error state;
  Temp_Fault:error event {Occurrence⇒Poisson λ1};
  Perm_Fault:error event {Occurrence⇒Poisson λ2};
  Recover:error event {Occurrence⇒Poisson λ3};
  Repair:error event {Occurrence⇒Poisson λ4};
  Detect:error event {Occurrence⇒Poisson λ5};
  NonDetect:error event {Occurrence⇒Poisson λ6};
  PerceiveFail:error event {Occurrence⇒Poisson λ7};
  H_Err:out error propagation {Occurrence⇒fixed λ8};
  H_FailedVisible:out error propagation {Occurrence⇒fixed λ9};
  H_OK:out error propagation {Occurrence⇒fixed λ10};
end CPUEM;
error model implementation CPUEM.impl
transitions
  Error_Free-[Perm_Fault]→ PermErr;
  Error_Free-[Temp_Fault]→ TempErr;
  TempErr-[Recover]→ Error_Free;
  PermErr-[Detect]→ Failed;
  PermErr-[NonDetect]→ ErrND;
  ErrND-[PerceiveFail]→ Failed;
  Failed-[Repair]→ Error_Free;
  TempErr-[out H_Err]→ TempErr;
  Failed-[out H_FailedVisible]→ Failed;
  Error_Free-[out H_OK]→ Error_Free;
end CPUEM.impl;
```

图 6 CPU 的错误模型

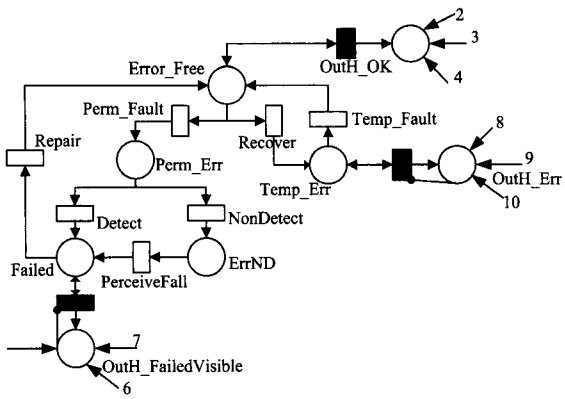

图 7 CPU 的 GSPN 模型

结束语 本文提出了一种基于 AADL 进行中断控制系统建模的方法,并利用基于 GSPN 的可靠性计算模型对可靠性进行分析。通过将中断控制器与 AADL 相结合,所提方法不仅弥补了 AADL 在中断控制建模中的不足,而且中断控制器的使用减少了 CPU 的负担,CPU 按照中断控制器提供的中断信号与中断向量码便可跳转至中断服务程序入口地址开始执行,从而避免了 CUP 参与中断优先级判决。

参 考 文 献

[1] Feiler P H, Gluch D P, Hudak J J. The Architecture Analysis & Design Language: An Introduction [R]. Carnegie Mellon University,2006

[2] SEI AADL Team. An extensible Open Source AADL Tool Environment(OSATE)[R]. SEI Carnegie Mellon University,2004

[3] Sokolsky O, Lee I, Clarke D. Schedulability analysis of AADL models[C]//Proc. 20th Int. Parallel and Distributed Proceedings Symposium,2006. USA: IEEE,2006:164-172

[4] 刘倩,桂盛霖,李允,等. 基于 UPPAAL 的 AADL 模型可调度性验证[J]. 计算机应用,2009,29(7):1820-1824

[5] 汤小明,苏罗辉,宋科璞. 飞行管理系统 AADL 建模与分析[J]. 计算机技术与发展,2010,20(3):191-194

[6] 许凌权,冯金富,左伟,等. 基于 AADL 的武器控制系统性能验证方法[J]. 电光与控制,2010,17(6):77-96

[7] 王庚,周兴社,张凡,等. AADL 模型的测试方法研究[J]. 计算机科学,2009,36(11):127-130

[8] 贾璐,胡林平,田丹. 基于 AADL 的航空电子系统安全性分析[J]. 航空计算技术,2009,39(5):58-61

[9] 谯婷婷,王乐,耶国栋. 基于 AADL 的软件可靠性验证[J]. 计算机应用,2012(s2)

[10] 李振松,顾斌. 基于 AADL 的中断控制设计方法[J]. 微型机与应用,2011,30(10):83-86

[11] 杨志斌,皮磊,胡凯,等. 复杂嵌入式实时系统体系结构设计与分析语言:AADL[J]. 软件学报,2010,21(5):899-915

[12] 董云卫,王广仁,张凡,等. AADL 模型可靠性分析评估工具[J]. 软件学报,2011,22(6):1252-1266